Amendment to the Specification

Please replace the paragraph beginning on page 2, line 15, with the following rewritten paragraph:

Whether code is compiled manually or through a series of questions, the conventional methodology for code generation is through use of a text editor and, after text has been edited, assembling that code into an object or execution file for use by the execution unit. While text is easily understood by the human editor and thus files in text format are beneficial, generating code from a text editor is sometimes cumbersome and time consuming. Also, it is not ~~easily determined by the~~ always easy for a user ~~of~~ to determine the relationship between his/her edits made to the text editor and the resulting, compiled code. ~~It~~ Therefore, it would be desirable to derive a less cumbersome and less time-consumptive process of generating execution code. ~~The~~ A desired process ~~should also~~ would be one which ~~can allow~~ allows the user to easily and quickly edit text using a more recognized Graphical User Interface ("GUI"), and to visually track the changes made using the GUI to ~~resulting~~ resultant changes in a source file. These benefits are currently lacking in conventional code generation techniques employing text editors.

A1

Please replace the paragraph beginning on page 3, line 3, with the following rewritten paragraph:

The problems outlined above are in large part solved by a code generation technique using links placed within corresponding lines of text designated as comments of a program. Specifically, a program can be displayed on a display device in, for example, a windows-based format. Given that windows-based programs are graphic intensive, the links can be accessed by a pointer device connected to a computer which executes the windows-based program. By directing the pointer device to the link contained in the same line as a comments designator of the program, the link can be activated by clicking the pointer device, or otherwise actuating the pointer device. Once activated, the link may have associated pull-down menus for changing the link in accordance with a selection made by the pointer device. Whenever the value shown by the link is changed or modified, a corresponding field within a source data set is also changed.

A2

Please replace the paragraph beginning on page 4, line 1, with the following rewritten paragraph:

According to one exemplary use, the links can be accessed through a GUI for programming an interface between two sub-systems. There are numerous types of programmable interfaces or "programmable devices." Generally speaking, a programmable device is any device that can be configured to achieve a specific logic function dependent on a software program. In other words, the logic device can be designed generically, and thereafter reconfigured through software to achieve a specific interface function. A popular programmable device can include gate arrays, logic arrays, and interface devices, each of which may be reconfigured in the field. The interface device can originally be designed as a general purpose interface device, which can later be reconfigured to achieve a specific purpose based on how that interface device is programmed. A General Purpose Interface Device ("GPIF") can be programmed from the source or data set derived from ~~alternations~~ alterations made to the present links. By changing the links within the comments portion of a program, a user can modify the data set (alternatively known as waveform descriptors in GPIF language) and then apply that data set to the execution unit of the GPIF. Thus, the data sets can be used by any execution unit, or by a particular execution unit applicable to a GPIF. Regardless of its application, the data sets can be easily modified and visually monitored within the same display window as the modifiable links.

A3

Please replace the paragraph beginning on page 13, line 23, with the following rewritten paragraph:

Fig. 6 illustrates the relationship between a comments portion of a program and the corresponding data set fields which, in this example, are known as waveform descriptors. The comments portion 80, and waveform descriptors 82 are similar to the comments portion 38 and data set portion 40 of Fig. 3. However, the primary difference is that the embodiment shown in Fig. ~~2~~ 6 is applicable to a GPIF compiled instructional code, and the data set of Fig. 3 is generic to any application.

A4

Please replace the paragraph beginning on page 14, line 11, with the following rewritten paragraph:

The layout of the comments portion 80 and the waveform descriptors 82 can apply to any structure desired, and the example provided is only one of various ways in which to note behavior, timing, and output descriptions of a waveform. Thus, waveform 3 ("Wave 3") can be described in numerous ways, with the example of Fig. 6 indicating three different characteristics placed in three different description subcategories noted as length/branch ("Len Br"), "Op Code," and "Output." Each interval can have different selectable links and corresponding comments. In the example shown, the comments note a particular address and data mode as either the same value or next value address or the data mode being no data or activated data. Next data and interface/wait comments will note whether the next data will be updated or whether the data will be the same as the previous data and whether the GPIF will be stalled (i.e., waited one, two, three, or more interface clock cycles), or if an interface will occur. Depending on whether the interval is in an DP state, various terms, such as Term A and Term B, can be combined according to a function (and/or/XOR) known as "LFunc." A DP interval can be repeated or re-executed if the re-execute comment contains a link in that interval to activate the re-execute mode. Control outputs can also be modified to either 1 or 0 states, as shown. The control outputs are needed to control the data and address signals forwarded from the GPIF to the peripheral device. If, in an DP interval, the LFunc is an "~~and~~AND" and Terms A and B are at a logic high state, then the DP interval will transition to the next interval only when the ~~anded~~ ANDed terms are met. Accordingly, if the ready input (RDY) is ~~anded~~ ANDed with another ready input (RDY), then DP will transition to the next interval only when the ready input (RDY) becomes active. Otherwise, DP may continue in, for example, its re-execute state if re-execute is toggled on through a link in the comments portion 80. For sake of brevity, Term A, Term B, LFunc, and Re-execute links are not shown since the setup of Fig. 6 is indicative of a NDP interval, whereby data is read by the GPIF or written from the GPIF in interval 1 due to the data mode being activated. The details of the various text links and corresponding comments, and the data set of corresponding fields is not applicable to the overall concept of simply having links which can be activated and modified by a on-screen pointer, and corresponding fields within certain command lines of a data set which are modified accordingly.